# STORAGE EFFICIENT MINIMIZATION LOGIC

## BACKGROUND OF THE INVENTION

The present invention relates to data processing techniques for use in communications systems and the like, and more particularly to the optimization of binary tree search structures used in such systems.

Determination of a minimum or maximum value in a set of data is a common operation which is often accomplished using specialized hardware. As an example, many communications systems utilize the Viterbi algorithm, which requires a determination of the most probable path for each state. This must be estimated by finding the path having the maximum metric (if correlation metric is used) or minimum metric (if Euclidean distance is used) among other concurrent paths. An efficient means for determination of the maximum or minimum metric is required.

A method for finding the minimum in an array of $N$ values which is storage efficient is to store the running estimate of the minimum and its location and compare it to each of the $(N-1)$ other values in the array. This method requires $(N-1)$ comparisons which will take at least $(N-1)$ clock cycles.

For delay sensitive systems or for systems having fewer than (N-1) clock cycles to do the comparisons, a standard alternate approach is the binary tree search structure. In a binary tree search structure, each layer contains half as many comparators and storage elements as the previous layer, thus reducing the latency to $\log_2 N$ clock cycles and allowing for pipelining. The downside of such a structure is the addition of (N-1) comparators and storage elements.

For the foregoing reasons, there is a need for a method and apparatus which limits the latency inherent in systems using a running estimate and which reduces the number of storage elements used in a standard binary tree search structure.

## SUMMARY OF THE INVENTION

The present invention is directed at a method and apparatus for considerably reducing the storage needed in a binary tree search structure.

In a preferred embodiment, an array of values is searched to find the minimum using a binary tree search structure in a storage efficient manner. A plurality of decision units are grouped in a plurality of computation stages. The number of decision units in a computation stage at level $i$ is equal to $N/2^i$, $N$ being the size of the array. Each computation stage reduces by ½ the set of values likely to contain the minimum of the array. Each decision unit in a computation stage takes as its input a pair of data values plus partial addresses of each data value, stores the minimum of the two values and adds to its partial address one most significant bit indicating the local address of the selected data value within the pair of inputs. In this embodiment, each computation stage adds one more bit of address until the entire address is known.

In one embodiment of the present invention, a system for locating a specific value contained in an array of $N$ data values is used. The specific value is the result of a binary operation defined over the array of $N$ data values wherein each data value is $W$ bits wide. The system comprises a plurality of decision units grouped in

successive computation stages wherein each decision unit receives a pair of input values, each input value containing a data value and a partial address, and generates a value representative of a selected data value and the partial address of the selected data value. In this embodiment, the decision unit of the last computation stage contains the specific value.

In another embodiment, the present invention is related to an apparatus for obtaining information on a specific value within a pair of inputs, wherein each input contains a data value and a partial address of the data value, the apparatus comprising a binary operator which compares the data values and which generates as output a binary decision representative of a local address of the specific data value and a multiplexer which generates as output the specific data value along with its partial address based on the binary decision.

In yet another embodiment the present invention provides, in an array of N data values, a method of determining an address for a result. The result is the output of a binary operation defined in the array of data values, each data value having $W$ bits. The method comprises the steps of performing, at each computation stage $i$ of $\log_2 N$ computation stages, $N/2^i$ binary operations on the data values of $N/2^i$ pairs of input values. In this manner, each of the binary operations generates a binary decision representative of a local

address of a selected data value within the pair of input values and multiplexes, at each computation stage, each pair of input values to produce an output determined by the binary decision.

It is an object of the present invention to provide a method for finding a specific value in an array of data values, the method comprising the steps of grouping a plurality of decision units in a plurality of computation stages wherein the number of decision units in a computation stage at level $i$ is equal to $N/2^i$, $N$ being the size of the array, and processing the data values in each decision unit. A decision unit at a last computation stage determines the specific value.

The present invention can be used in any array having a transitive binary operator defined on it wherein the binary operator maps any pair to the Boolean set of {true, false}.

These and other features and objects of the invention will be more fully understood from the following detailed description of the preferred embodiments which should be read in light of the accompanying drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and form a part of the specification, illustrate the embodiments of the present invention and, together with the description, serve to explain the principles of the invention.

Figure 1 shows a logic circuit for performing the method of the present invention;

Figure 2 shows the result obtained by applying the method of the present invention;

Figure 3 shows memory savings obtained with respect to previous techniques when using the present method and apparatus.

## DETAILED DESCRIPTION OF THE INVENTION

In describing a preferred embodiment of the invention illustrated in the drawings, specific terminology will be used for the sake of clarity. However, the invention is not intended to be limited to the specific terms so selected, and it is to be understood that each specific term includes all technical equivalents which operate in a similar manner to accomplish a similar purpose.

With reference to the drawings in general and Figures 1 through 3 in particular, the apparatus of the present invention is disclosed.

Figure 1 shows a logic circuit for performing the method of the present invention, and illustrates a circuit for obtaining the minimum value contained in an array of data values along with its location address. In this embodiment, sixteen data values X15, X14,...,X0 contained in a memory area are searched to find the minimum value. Four computation stages, each having at least one decision unit, are used to determine the minimum and its location. Each computation stage prunes the number of possible values by one-half. In this embodiment, the possible values are the reduced set obtained from the array of data values by pruning half the values that are not the minimum values. The number

of decision units at each stage is equal to one-half the number of input values.

In this embodiment, each decision unit in a computation stage takes as its input a pair of data values plus partial addresses of each data value, stores the minimum of the two input values and adds to its partial address one most significant bit indicating which of the two data values was picked. Each computation stage adds one more bit of address until the address is completely known.

As illustrated in Figure 1, a decision unit is composed of a binary operator represented here as comparator 101, a multiplexer 103 and a storage element 105. Each comparator 101 at computation stage 100 takes as its input a pair of data values each eight bits long and outputs a binary decision representing the local address of the minimum data value within the pair. As an example, if X15 is the minimum compared to X14, the binary decision will be a binary 1 since "1" is the local address of X15 and "0" is the local address of X14 within the pair.

A multiplexer 103 at first computation stage 100 takes as its input a pair of values from the array of data values and outputs the input value selected by the binary decision. In this embodiment, the binary decision designates which input value corresponds to the Boolean "true" of the binary operator, and thus to the minimum of

the two input values. A storage element 105 at first computation stage 100 stores the output of its associated multiplexer 103.

In one embodiment, the storage element 105 stores the selected data value in one section of memory storage and adds a local address bit represented by the binary decision as its most significant bit to the partial address of the selected data value. Alternatively, the local address can be added as the least significant bit of storage element 105 and the selected data value can be the eight most significant bits.

In this embodiment, at first computation stage 100, the partial addresses of the data values are 0 bits wide. At first computation stage 100, each storage element 105 contains a selected data value and one bit representing the partial address of the selected data value. The set of data values contained in all storage elements of first computation stage 100 represents the reduced set of data values which are likely to contain the minimum of the array of data values. The minimum value of the array of data values is also referred to as a specific value according to the binary operation defined over the array of data values.

At second computation stage 110, each multiplexer 113 takes as its input the content of a pair of storage elements 105 of first computation stage 100. As an example, the inputs of multiplexer 113a are the content

of storage element 105a and 105b. In a preferred embodiment, the inputs of a decision unit are the two consecutive storage elements' contents of the previous computation stage. The inputs of the multiplexer 113 are each nine bits wide and represent an eight bit wide selected value plus one bit of partial address. In a preferred embodiment, the comparator 111 compares only the eight least significant bits of the input values to determine the minimum and outputs a binary decision. Storage element 115 stores the output of multiplexer 113 selected by the binary decision and adds the binary decision as the most significant bit to the partial address. At second computation stage 110, the possible values are reduced to a set of four elements.

The operation previously described in accordance with first computation stage 100 and second computation stage 110 is performed at each subsequent computation stage. At third computation stage 120, the decision units' inputs are ten bits wide while at fourth computation stage 130 they are eleven bits wide. The last stage, represented by fourth computation stage 130, contains only one decision unit, as illustrated in Figure 1. The storage element 135 contains four bits representing the entire address of the specific value which is stored in the eight least significant bits portion of the storage element 135.

Figure 2 illustrates the result obtained by applying the method of the present invention to an array of values. At each computation stage, the content of the storage elements is shown. The last computation stage contains the minimum value which is equal to one and its entire address which in binary format is 0101.

Although Figure 1 shows an array of values of size sixteen, the present invention can be applied to any array of values having a size $N$ wherein $N$ is a power of two and each value is $W$ bits wide. The number of computation stages in this case is equal to $\log_2 N$. Each computation stage at level $i$, with $1 \leq i \leq \log_2 N$, contains $N/2^i$ decision units and each decision unit takes two inputs, each $(W+i-1)$-bit wide.

The method of the present invention can also be applied to any set which has a binary operation defined on it and wherein the binary operation has a transitive property. In particular, the method can also be used to find a maximum in an array of values.

The curve of Figure 3 shows the number of storage elements saved over the prior art when using the method and apparatus of the present invention. In the prior art binary tree search structure, the values along with their entire address are stored at each stage of the search, whereas with the present method only partial addresses are stored. The number of bit storage elements (NUM FLOPS) needed in the prior system is represented by the

following formula: $W(N-1)+\log_2 N(N-1)$. The number of bit storage elements needed for the present system is equal to $W(N-1)+2N-\log_2 N-2$. The overall storage savings can then be predicted by the formula: $N\log_2 N-2N+2$ which is represented by the curve of Figure 3.

Although the invention has been illustrated by reference to specific embodiments, it will be apparent to those skilled in the art that various adaptations and modifications may be made which clearly fall within the scope of the invention. The invention is intended to be protected broadly within the spirit and scope of the appended claims.